Event collections and datasets

David Adams
November 8, 2002

Introduction

LCG POOL has a "Collections and Metadata" work package intended to support ensembles of POOL objects. See http://lcgapp.cern.ch/project/persist/metadata. A significant amount of related work has been done in the BNL dataset project described at http://www.usatlas.bnl.gov/~dladams/dataset. Here we examine some of the commonalities and differences between these models and make some proposals for integration.

Commonalities

Both models provide means to describe a container of objects and iterate over the objects in the container. Both are generic in that they can hold objects of any type but the major use case for both is to hold HEP event data. When we say that a collection/dataset holds a data object, this generally means that it holds a global identifier that can be used in conjunction with a persistent store (e.g. POOL) to locate the object.

Both envision both explicit and implicit collection/datasets. In an explicit collection/dataset the objects (or more precisely their identifiers) are held directly. An implicit collection/dataset holds the objects (or a subset of the objects) held in some other container (e.g. a file or another collection/dataset).

Features in Collection but not in Dataset

Tag data

The collection model envisions that optional tag data is associated with each object in the collection and that queries using this data can be used to select objects. This tag data would normally be stored in a relational database.

The dataset model does not explicitly support tag data except for the local identifiers used to label objects within a dataset. Both models support queries based on the data in the collection/dataset and the dataset model does allow some of the contained data to be held in relational database.

Event headers

The collection model envisions that the data in an event is described by an event header that holds a list of identifiers for the data objects associated with the event. The header is itself an object and the collection holds identifiers for the included event headers.

The dataset directly holds the identifiers for the event data objects. It is allowed for the dataset to also hold event headers. Even if these we not explicitly held, the nature of the dataset makes it possible to construct them on demand if desired.

Features in dataset but not collection

Local identification (indexing)

A dataset is a map of data objects indexed by a local identifier that is unique in the context of the dataset but not unique or necessarily recognizable in the context of the persistent. The dataset provides the connection between the local identifier and the global identifier used to locate the object.

For an event header (if they exist), the local identifier is an event ID, e.g. run and event number. For a data object in an event, the local identifier is an event ID plus a "content ID" that distinguishes between the different types of data in an event. For ATLAS, we might follow the transient model and construct the content ID from the type (class ID) and string key used in StoreGate.

In contrast, an explicit collection is simply a vector of global identifiers. For an event collection these would be identifiers for event headers. The event ID would be included in one of the data objects referenced by the event header (e.g. the EventInfo object). Each data object might similarly carry its event and content ID's.

The indexing inherent in a dataset makes it possible for a dataset to report on its content without ever locating or opening the data objects that it holds. E.g., an event dataset can report which events are held (i.e. for which event ID's) and the content (list of content ID's) for any of these events. A consistent event dataset would have the same content for each event and one can speak of the content of the dataset.

For explicit collections, event-specific properties such as the event ID and content could be included in the tag database. Collective properties such as the event ID list and the content could be stored in a collection catalog.

File association

The dataset model envisions making an association between a dataset and the set (or sets) of files holding the data in the dataset. This information allows a job scheduler to gather the required files or to send the job to a location where the files are easily accessible. This information might be added to a catalog in the collection model.

Content merging

Event datasets support the merging of datasets with different content for the same events. E.g. the jets reconstructed in one job could be merged with the tracks reconstructed in another. More importantly, a single job could write tracks and jets to separate datasets and then trivially create a third dataset merging the content in these two. Presumably this is done in the explicit collection model by creating a third set of event headers.

Content Selection

The event dataset model allows for selection based on content, i.e. to restrict the content in an existing dataset to create a new dataset. The main advantage is to reduce the number of files required where the data is distributed over multiple files but there are also cases where one would like to hide existing data because it is going to be replaced with new data with the same content ID.

Portability

Datasets are portable, i.e. they have a persistent representation that can be written into a file and copied from one location to another. The current implementation uses XML. The collection model is based on catalogs but one could presumably add a portability layer that extracts the relevant data.

Non-event data

The dataset model allows non-event data to be included in a dataset with event data. For example relevant conditions data might be included. Collections iterate over a single type and do not naturally accommodate non-event data.

Other issues

Replicated object identity

An object may be replicated, i.e. copied from one physical container (e.g. file) to another. Does the object retain its identity in the new container or is it assigned a new identity? If so, then references to the object can be satisfied with either container and do not have to be updated to refer to the new copy. If not, then the new identifier can carry an identifier of the new container to add in locating the object. The existing collection model assumes the identity changes while the dataset model assumes it retains the old identity. Although either choice can be made to work, it would be good to resolve this issue soon.

Object references

Do references to another object use a global identifier as assumed in the previous section? Or are they allowed to use a local identifier and depend on context (collection/dataset) for resolution?

Collection Issues

Implicit collection tags

Should implicit collections include tags?

Separating tags and collections

Should tags be separated from collections and indexed with event ID's so that a selection can be done on one collection and then applied to another?

Integration

Here we consider some strategies for integrating the event collection and dataset models. Event collections are already part of the LCG project.

Ignore datasets

One option is to ignore all the new capabilities in datasets and stick with the existing collection model.

Absorb datasets

Many or all of the features of the dataset model are desirable. The collection model could be expanded to absorb them as discussed above.

Datasets as implicit collections

The collection model is mostly devoted to explicit collection while many of the dataset features are relevant to explicit collections. It is natural to expand the current narrow range of implicit collections to encompass all or most of the range of dataset varieties.

This might include adding a thin dataset layer over the explicit event collection so that collection/dataset users have a common interface for accessing data from implicit and explicit containers.